StickyLocalization: Robust End-To-End Relocalization on Point Clouds using Graph Neural Networks

Kai Fischer[1,3] Martin Simon[1,3] Stefan Milz[2,3] Patrick Mäder[3]
[1]Valeo Schalter und Sensoren GmbH [2]Spleenlab GmbH [3]Ilmenau University of Technology
{kai.fischer, martin.simon}@valeo.com {stefan.milz, patrick.maeder}@tu-ilmenau.de

Abstract

Relocalization inside pre-built maps provides a big benefit in the course of today's autonomous driving tasks where the map can be considered as an additional sensor for refining the estimated current pose of the vehicle. Due to potentially large drifts in the initial pose guess as well as maps containing unfiltered dynamic and temporal static objects (e.g. parking cars), traditional methods like ICP tend to fail and show high computation times. We propose a novel and fast relocalization method for accurate pose estimation inside a pre-built map based on 3D point clouds. The method is robust against inaccurate initialization caused by low performance GPS systems and tolerates the presence of unfiltered objects by specifically learning to extract significant features from current scans and adjacent map sections. More specifically, we introduce a novel distance-based matching loss enabling us to simultaneously extract important information from raw point clouds and aggregating inner- and inter-cloud context by utilizing self- and cross-attention inside a Graph Neural Network. We evaluate StickyLocalization's (SL) performance through an extensive series of experiments using two benchmark datasets in terms of Relocalization on NuScenes and Loop Closing using KITTI's Odometry dataset. We found that SL outperforms state-of-the art point cloud registration and relocalization methods in terms of transformation errors and runtime.

1. Introduction

Continuous allocation of a vehicle's ego pose is essential for today's autonomous driving tasks. Being the prerequisite for subsequent tasks, such as trajectory planning or accomplishing complex driving maneuvers, it is necessary to provide highly precise estimations of a vehicle's position in sufficient intervals facilitating real-time capability of the driving systems. Global Navigation Satellite Systems (GNSS) or Real Time Kinematic (RTK) aided systems are widely used to tackle these problems as they are able to supply pose information with an accuracy of a few centimeters. Due to fluctuating connection quality of GPS systems, e.g., tall buildings in urban environments blocking the signal, an accepted approach is incorporating pre-built maps into the pose estimation pipeline. These maps can for instance consist of accurate geometrical modeling of the environment like high definition maps (HD-Maps) or measurements of the vehicle's on-board sensors.

In order to utilize these maps in a pose estimation task, it is necessary to establish correspondences between current sensor data and the map's content to calculate a transformation relative to a reference coordinate frame. A common way of accomplishing this task, e.g., with the aid of LiDAR (Light Detection and Ranging) scanners, is applying point cloud registration approaches, like iterative closest points (ICP), to the current point cloud and a map section close to an initial location guess as provided by a low cost GPS. However, these systems are only able to deliver an insufficient position accuracy of 5–20 m letting traditional as well as up-to-date point cloud registration methods often fail. Thus, explicit relocalization approaches, robust against large translational drifts as well as big rotational offsets in the pose initialization and still being capable of providing accurate pose estimations with low computation time, are urgently needed. Besides Simultaneous Localization and Mapping (SLAM) and odometry estimation methods, a comprehensive review of deep neural network (DNN) relocalization approaches was done by [5]. The authors found that main research focus in the topic lays on image based methods yielding 2D-to-2D and 2D-to-3D localization while 3D-to-3D being comparably under-explored to the date. We argue that cameras are highly sensitive to insufficient and varying illumination resulting in relocalization drifts and errors and making research into LiDAR-based methods a timely endeavor. In this paper, we present a novel and fast end-to-end deep learning approach for accurate single-shot map relocalization on point clouds utilizing graph neural networks. Our main contributions are as follows:

- a novel end-to-end graph neural network-based solution

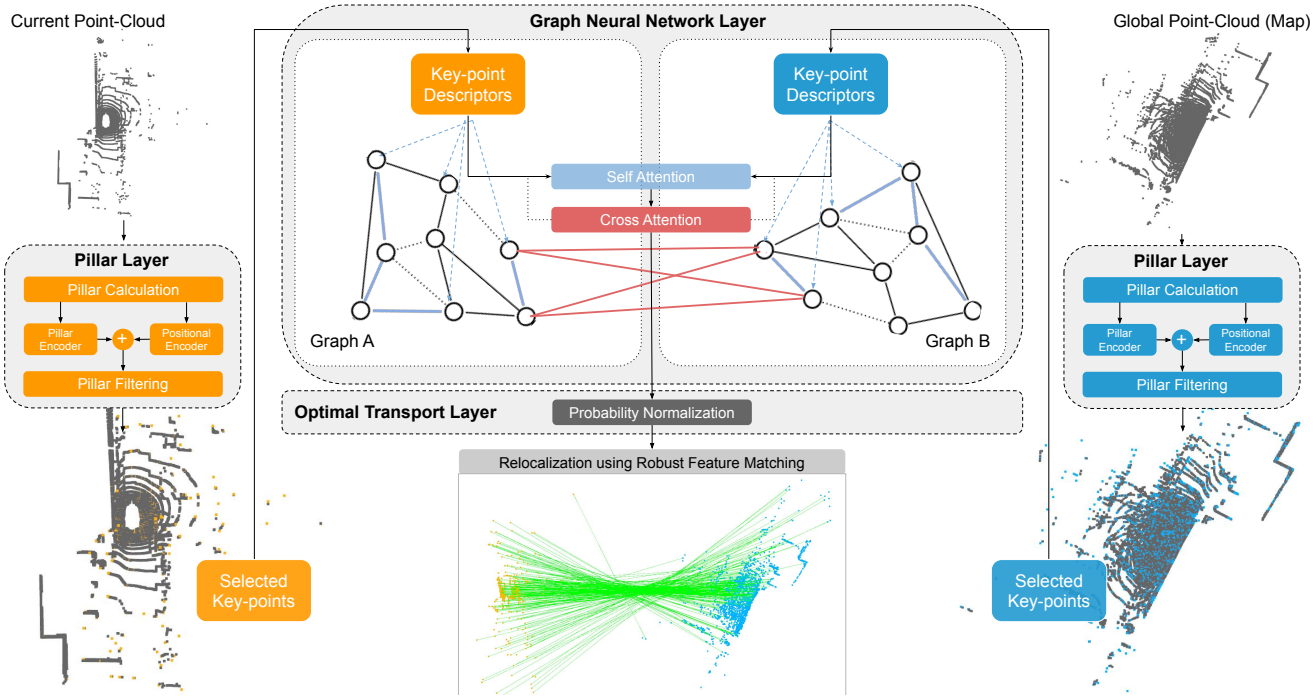

Figure 1. **Overview of the StickyLocalization architecture** consisting of the pillar construction and encoding by the *Pillar Layer*, feature point detection (orange and blue) through *Pillar Filtering*, context aggregation inside the *Graph Neural Network Layer* and probability normalization by the *Optimal Transport Layer*.

for relocalization and loop closing tasks yielding low computation time without the necessity of pre-selected key-point features;

- a novel distance based matching loss improving network convergence for correspondence matching tasks;
- extensive experiments on the nuScenes and KITTI Odometry dataset demonstrating StickyLocalization's robust performance in relocalization and loop closing applications; and
- a comparison of StickyLocalization's performance to state-of-the-art methods for point cloud registration and relocalization employing large translational and rotational offset scenarios.

2. Related Work

Relocalization methods mostly utilize pre-built maps to estimate the vehicle's poses based on current sensor measurements. Therefore, correspondences of current scan points have to be determined in the complete map (global) or sections of it (local).

Local Relocalization approaches aim to determine the vehicle's pose based on a subsection of the map showing its near proximity based on an inaccurate initial guess acquired by low performance inertial systems. In this context point cloud registration methods can be utilized to solve the problem of map relocalization and loop closing in the course of LiDAR odometry and mapping approaches [23, 32, 14]. The most prominent approach in this domain is ICP and it's adaptions [3, 33, 18] where features are matched to their particular correspondences by iteratively minimizing the overall average point distance. These methods, however, often crucially depend on a sufficiently close initialization and high overlap of succeeding point clouds. Another common approach is employing a 2D or 3D grid representation of the map consisting of different features like Gaussian distribution of LiDAR reflectances [12, 13] or Gaussian Mixture Models in height-direction of the laser scanners [29]. At inference time, current scans are processed in the same manner to establish feature based map matching. [6] propose a different approach by sliding a window to minimize the distance between the current features and the ones of the closest map frames.

Recently, the emergence of deep neural networks inspired point cloud registration and relocalization methods respectively. [28] for example combines transformer networks with a subsequent SVD network structure to estimate rigid transformations based on learned feature representations to an end-to-end structure. Experiments show highly accurate pose estimations compared to state of the art methods but are limited to synthetic point clouds and the presence of low drifts (< 0.5m) and thus is not applicable in terms of autonomous driving tasks. In contrast, [15] utilizes a DNN

based approach for outdoor scenarios by predicting corresponding points in the other cloud to estimate the rigid transformation. However, similar to common ICP approaches a decent initial estimate is required. Methods like [30] and [2] utilize a joint learning process of feature detection and description where salient points of point clouds are simultaneously determined and equipped with significant descriptors subsequently used to find correspondences and estimate the rigid transformation. On the other hand [25] utilizes a DNN based pre-filtering of point clouds to remove dynamical and temporary static objects like driving or parking vehicles. Based on the consequential static clouds distinctive points are extracted from current scans for a robust map feature matching and pose estimation. [7] uses pre-processed point cloud features in a PointNet [17] like structure for a direct 2D-Pose estimation with subsequent extended Kalman filters yielding a real time relocalization application.

Global Relocalization methods in contrast to local ones determine the vehicle's pose inside the pre-built map without the need for an initial guess. In [1] and [31], a DNN is used to learn global feature descriptors which can be used to measure the similarity between point clouds. Utilizing high dimensional kd-trees based on these descriptors, the closest map frame for an initial position guess can be determined. Subsequently, common approaches like particle filters and ICP are used to refine the pose estimation. [10] and [21] utilize image based features retrieved by a customized GoogLeNet [24] and *Graph Neural Networks* respectively to determine global correspondences for a six DOF pose estimation. In contrast, [27] is able to predict a six DOF pose inside a map by feeding a single point cloud frame to the network. This is achieved by implicitly learning a representation of the complete map in the course of the training process. However, such relocalization methods are consequently restricted to the scene employed in the training process and are therefore not generalizable to multiple environments.

3. The StickyLocalization Architecture

StickyLocalization aims to provide fast and robust 3D-to-3D (local) relocalization for autonomous driving allowing for accurate pose correction even based on a potentially bad initialization as expected from low cost GPS devices.

Problem description: We define $\mathcal{P}^{\mathrm{G}}$ and $\mathcal{P}^{\mathrm{L}}$ as two point clouds to be registered, whereas L defines the local point cloud and G global map. The key-points of those point clouds will be denoted as $\boldsymbol{\pi}_i^{\mathrm{G}}$ and $\boldsymbol{\pi}_j^{\mathrm{L}}$ with $\{\boldsymbol{\pi}_0^{\mathrm{G}}, \dots, \boldsymbol{\pi}_n^{\mathrm{G}}\} \subset \mathcal{P}^{\mathrm{G}}$ and $\{\boldsymbol{\pi}_0^{\mathrm{L}}, \dots, \boldsymbol{\pi}_m^{\mathrm{L}}\} \subset \mathcal{P}^{\mathrm{L}}$, while other points will be defined as $\boldsymbol{x}_g^{\mathrm{G}} \in \mathcal{P}^{\mathrm{G}}$ and $\boldsymbol{x}_l^{\mathrm{L}} \in \mathcal{P}^{\mathrm{L}}$. Every *point pillar* is associated with a key-point index i, a cylinder with an endless height. The pillar is encoded by a feature descriptor with depth D. Only most significant pillars, selected by the Filtering Layer, will be used as graph neural network input. The overall goal is to find partial assignment of two graphs composed by specific feature descriptors, i.e. point pillars.

To keep computation time low and being independent to point cloud pre-processing for feature extraction, a joint learning process for key-point detection and matching comprises reasonable advantages. Therefore we propose an end-to-end trainable approach for combined key-point selection and matching on raw LiDAR point clouds utilizing *Graph Neural Networks* for geometrical context aggregation. The subsections below describe the architecture of our approach organized according its main components: (1) the *Pillar Layer*, (2) the *Graph Neural Network layer*, and (3) the *Optimal Transport layer*.

3.1. The Pillar Layer

The pillar layer is comprised of three core components: a pillar encoder, positional encoder and pillar filtering. This is a very important clue, the pillar encoder calculates the pillars for the whole cloud, whereas the positional encoder is responsible for geometric aggregation. The pillar filtering selects the most distinctive features to be matched.

Pillar Encoder: We learn features in 3D, i.e. identified by key-points, being inspired by [11], because it is important to get a local strong descriptor of the environment. A key point, $\boldsymbol{\pi}_i^{\mathrm{G}}$ and $\boldsymbol{\pi}_j^{\mathrm{L}}$, relates to a *point pillar* i and j including a set of specific points $\mathcal{P}_i^{\mathrm{G}}$ and $\mathcal{P}_j^{\mathrm{L}}$. Sampling is realized using euclidean distance in x,y-plane with infinite height, resembling a pillar shape. The *pillar encoder*'s feature input stack $\mathbf{f}_i^{\mathrm{G}} \in \mathbb{R}^D$ (with D as input feature depth) per sampled point $\boldsymbol{x}_\Omega^{\mathrm{G}}$ with $\Omega \in \{1, \dots, z\}$ is composed as:

$$\mathbf{f}_i^{\mathrm{G}} = \left\{ \left[(\boldsymbol{x}_\Omega^{\mathrm{G}} - \boldsymbol{\pi}_i^{\mathrm{G}}), i_\Omega^{\mathrm{G}} \right], \dots \right\} \tag{1}$$

with $\boldsymbol{x}_\Omega^{\mathrm{G}} \in \mathbb{R}^3$ denoting the $(x, y, z)^{\mathrm{T}}$ coordinates and $\boldsymbol{i}_\Omega^{\mathrm{G}} \in \mathbb{R}$ being a scalar intensity value (e.g. LiDAR reflectance). In contrast to [11] and [8] which are calculating a 11 dimensional vector for each point inside the pillar, we found that the respective 3D coordinates relative to the pillar's keypoint $(\boldsymbol{x}_\Omega^{\mathrm{G}} - \boldsymbol{\pi}_i^{\mathrm{G}})$ + intensity are sufficient for describing the *local* context of the feature point. More import in this regard is the selection of the shape of the local neighbourhood for feature aggregation where we found the pillar shapes to be superior to compared ones like voxels. Experiments on this are given in the supplementary material. The pillar encoder itself is a single linear projection, sharing weights across all pillars and frames, it is preceded by batch normalization and eventually ReLU activated. The output is defined by: $\mathbf{f'}_j^{\mathrm{G}}, \mathbf{f'}_i^{\mathrm{G}} \in \mathbb{R}^{D'}$:

$$\begin{aligned} \mathbf{f'}_i^{\mathrm{K}} &= \mathrm{W}_{\mathbf{f}} \cdot \mathbf{f}_i^{\mathrm{G}} \quad \forall i \in \{1, \dots, n\} \\ \mathbf{f'}_j^{\mathrm{L}} &= \mathrm{W}_{\mathbf{f}} \cdot \mathbf{f}_j^{\mathrm{L}} \quad \forall j \in \{1, \dots, m\} \end{aligned} \tag{2}$$

Positional Encoder. The positional encoder analyzes and captures the geometrical *global* context of a pillar [17].

It utilizes a multi-layer-perceptron (MLP) such as all pillars with preceding batch normalization and ReLU activation. The weights are shared across $\mathcal{P}^{\mathrm{L}}$ and $\mathcal{P}^{\mathrm{G}}$. From the key-points coordinates $\boldsymbol{\pi}_i^{\mathrm{G}}, \boldsymbol{\pi}_j^{\mathrm{L}} \in \mathbb{R}^3$, we derive the positional features as:

$$\begin{aligned} \boldsymbol{\pi}'_{i^{\mathrm{G}}} &= \mathrm{MLP}_{\boldsymbol{\pi}}(\boldsymbol{\pi}_i^{\mathrm{G}}) \quad \forall i \in \{1, \dots, n\} \\ \boldsymbol{\pi}'_{j^{\mathrm{L}}} &= \mathrm{MLP}_{\boldsymbol{\pi}}(\boldsymbol{\pi}_j^{\mathrm{L}}) \quad \forall j \in \{1, \dots, m\}\,. \end{aligned} \tag{3}$$

Finally, we combine the aggregated local and global context information of the pillars into the final feature representation:

$$\begin{aligned} \boldsymbol{n}_i^{\mathrm{G}} &= {\pi'}_i^{\mathrm{G}} + {\mathbf{f}'}_i^{\mathrm{G}} \\ \boldsymbol{n}_j^{\mathrm{L}} &= {\pi'}_j^{\mathrm{L}} + {\mathbf{f}'}_j^{\mathrm{L}} \qquad \boldsymbol{n}_i^{\mathrm{G}}, \boldsymbol{n}_j^{\mathrm{L}} \in \mathbb{R}^{D'} \end{aligned} \tag{4}$$

Here we found, that it is very important to have a separated processing of the local and global information by the pillar- and positional-encoder respectively. For instance combined aggregation by using the points' global 3D coordinates as the pillar encoder's feature input stack, instead of the relative ones and omitting the positional encoder pipeline, would lead to the network not converging at all.

Pillar Filtering: We adapt descriptor based key-point feature selection in our joint training process similar to [2]. We propose a hard key-point selection strategy for both, training and testing, since we found that this allows the network to derive a more robust point representation. Furthermore instead of choosing the most salient entry of the feature descriptors for $\boldsymbol{n}_i^{\mathrm{G}}$ and $\boldsymbol{n}_j^{\mathrm{L}}$ as proposed by [2], we calculate the euclidean norm of each point's D' dim feature descriptor, which leads to a more robust selection strategy and a smoother training procedure.

Subsequently, this value is verified to be the maximum in the context of the local spatial neighborhood N_{x_i} within a certain radius r surrounding the considered point. Eventually, key-point candidates are chosen from among all points satisfying the above criteria by selecting the n and m most salient points for the respective clouds. We furthermore found that normalizing the feature descriptors by the number of points in the respective local neighborhood is helping to stabilize the training process, guaranteeing a more homogeneous point selection strategy.

3.2. Graph Neural Network Layer

The input nodes of the graph (Layer 0) are the descriptors of the filtered key-points for the local ${}^{(0)}\boldsymbol{n}_i^{\mathrm{L}}$ and global ${}^{(0)}\boldsymbol{n}_i^{\mathrm{G}}$ cloud respectively. The **Graph Architecture** is inspired by [21] and [8] and developed in the same way. The idea relies on two complete graphs $\mathcal{G}^{\mathrm{L}}$ and $\mathcal{G}^{\mathrm{G}}$, whose input nodes are equivalent to the pillars (cp. Fig. 1). The graph is a *multiplex* graph to perform global matching with inter-frame edges, i.e. edges that connect all nodes of $\mathcal{G}^{\mathrm{G}}$ with $\mathcal{G}^{\mathrm{L}}$ and vice versa. The extensive use of multi-head self attention, for each graph individual, as well as multi-head cross attention, among both graphs cross connected, effectively integrates contextual cues. We propose an attention function $\mathcal{A}$ [26] with query q, keys k, and values v, defined as:

$$\mathcal{A}(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}) = \mathrm{softmax}\left(\frac{\boldsymbol{q}^{\mathrm{T}} \cdot \boldsymbol{k}}{\sqrt{D'}}\right) \cdot \boldsymbol{v} \tag{5}$$

D' is the feature depth of each nodes input and output (see [8]).

After applying several layers $\mathrm{l}_{\max}$ similar to [8] the graph ends up in **final predictions** computed by the last layer and designed as single linear projection with shared weights across both graphs $(\mathcal{G}^{\mathrm{L}}, \mathcal{G}^{\mathrm{G}})$ and pillars:

$$\begin{aligned} \mathbf{m}_i^{\mathrm{K}} &= \mathrm{W}_{\mathbf{m}} \cdot {}^{(\mathrm{l}_{\max})}\boldsymbol{n}_i^{\mathrm{G}} \\ \mathbf{m}_j^{\mathrm{L}} &= \mathrm{W}_{\mathbf{m}} \cdot {}^{(\mathrm{l}_{\max})}\boldsymbol{n}_j^{\mathrm{L}} \end{aligned} \tag{6}$$

3.3. Optimal Transport Layer

Similar to [8] and [21] the final matching consists of two steps, first a score matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ is calculated based on the last layers output:

$$\mathbf{M} = (\mathbf{m}^{\mathrm{K}})^{\mathrm{T}} \cdot \mathbf{m}^{\mathrm{L}} \tag{7}$$

In a second step a soft-assignment matrix $\mathbf{P} \in \mathbb{R}^{(n+1)\times(m+1)}$ is containing matching probabilities for each pair of features is derived. As proposed by [21] we extend $\mathbf{M}$ to $\bar{\mathbf{M}}$ by using a dustbin for unassigned matches. Finding the optimal assignment then corresponds to maximizing the sum $\sum_{i,j} \bar{\mathbf{M}}_{i,j}$, as described in [8].

3.4. Loss

Previous point matching approaches, e.g., [21], propose a hard assignment of point correspondences between frames by element-wise multiplication of a ground truth match matrix $\mathcal{GT}$ with the predicted match probabilities $\mathbf{P}$ used for calculating the negative log-likelihood loss:

$$\mathcal{L}_{\mathrm{NLL}} = -\sum_{i,j \in \mathcal{GT}} \log \mathbf{P}_{ij}. \tag{8}$$

Thereby, the ground truth matrix comprises 1 for a correspondence match and 0 otherwise, with a match referring to an inter-frame nearest neighbor distance below a certain threshold for each point in the respective frames. This method exhibits good performance for networks with pre-selected features since it facilitates fast training convergence. However, using it for the joint learning process including key-point selection the we propose, this approach is prejudicial. This is due to close points with a distance slightly larger than the nearest neighbor and therefore not depicting the corresponding match point would be valued as 0, penalizing the learned representation of close and similar

points by raising the loss in the training process. Therefore, we propose a novel distance-based matching loss addressing joint training of key-point selection and matching. To reward closer and penalize farther points, the confidence values predicted by the network are multiplied with a weighing term based on the euclidean distances $\mathbf{D}$ of the respective points before calculating the negative log-likelihood loss, which smooths the training process and yields faster convergence in the joint training process.

$$\mathcal{L}_{\mathrm{NLL}} = -\sum_{i=0}^{m}\sum_{j=0}^{n}\log(\mathbf{P}_{ij}\cdot e^{-\mathbf{D_{ij}}}) \tag{9}$$

with m and n expressing the number of points in the respective frames.

4. Experiments

We evaluate StickyLocalization's performance from two perspectives. First, we evaluate the approach's relocalization capabilities under challenging conditions using the nuScenes dataset. We compare our results to classical as well as DNN-based state-of-the-art approaches for point cloud registration in terms of average transformation errors. Second, we evaluate how StickyLocalization can be adopted to tackle loop closure in LiDAR-based odometry and mapping. More specifically, we show how our approach can substitute common techniques for loop closure estimation like ICP and other state-of-the-art point cloud registration methods on selected sequences of the KITTI dataset. Finally we show the impact of the joint learning process for key-point detection and matching utilizing the proposed selection strategy and loss function with regard to the overall performance of our approach by comparing different versions of StickyLocalization.

4.1. Experimental Setup

Model configuration: Each *point pillar* is sampled with up to $z = 128$ points using a Euclidean distance threshold of $d = 0.5\,\mathrm{m}$. We utilize a feature depth of $D' = 256$. The positional encoder has five layers with the dimensions set to $32, 64, 128, 256$ channels respectively. For the key-point selection layer (Pillar Filtering) we chose the $m = 500$ and $n = 2500$ most salient candidates for the current frame and the accumulated map cloud respectively. The graph is composed of $l_{\max} = 9$ *self* and *cross* attention layers with $h = 4$ heads each. In depth experiments on the selection and comparison of hyperparameters, as well as other design choices of the architecture can be found in the supplementary material. Our model is implemented in *PyTorch* [16] *v1.8.1* with *Python 3.6*. If not stated otherwise, each training is performed using the Adam optimizer and the proposed distance based matching loss (cp. Eq. 9). Pose estimation is performed by applying singular value decomposition (SVD) of the predicted matches by the network with correspondence probabilities below a threshold of 0.2 being filtered. All experiments were executed on a system equipped with Intel Xeon E5-1650 CPU, NVIDIA GeForce GTX 1080 Ti GPU and 64GB of RAM.

Datasets: For validation on the relocalization task we utilize the nuScenes dataset [4], which consists of 1,000 driving scenes of challenging environments captured in Boston and Singapore. The dataset includes recordings of 6 cameras, 5 radar sensors, and a 32-layer LiDAR scanner. Apart from annotations for semantic segmentation and object detection, nuScenes also provides ground truth poses for each recorded frame obtained from a reference RTK system. Since the same scene has been recorded multiple times, the dataset perfectly suits our needs for evaluating relocalization performance under challenging conditions. Furthermore, we aim to study generalization ability and flexibility of our approach by applying it to loop closure estimation tasks. Therefore, we employ KITTI odometry's training sequences with overlapping trajectories offering the ability to perform loop closings with the provided Velodyne scans and ground truth poses.

Training details: For our trainings on nuScenes we utilize the *Singapore-Onenorth* environment training set consisting of multiple sequences captured on nine different days. In order to generate frame to map pairs, we process every frame and acquire corresponding map frames from sequences on different days within a $20\,m$ radius relative to the original shot using the provided poses. Maps containing an amount of consecutive frames larger than a predefined threshold, i.e. 5 in this case were considered valid. By applying the ground truth transformations to the current as well as the respective accumulated map frames, we gain source and target input point clouds to be used with our approach. To balance the variable data size of the input clouds, we perform voxelization with a voxel size of $0.1\,m$ on the source and with a voxel size of $0.5\,m$ on the target, i.e., the map cloud before feeding it to the network. In order to have a large variety in the training data and to ensure a robust and generalizable network, we use random translation within a range of $\delta_t = [0, 20]\,m$ and a random rotation of the yaw angle of $\delta_r = [-\pi, \pi]$ to generate the poses to be estimated by the network in the course of the training process. Since nuScenes provides 2D poses, in this context the selected δ_t are limited to a Euclidean distance for a translation in x and y direction $\delta_t = \sqrt{t_x^2 + t_y^2}$.

For loop closure estimation on the KITTI odometry dataset, we pursue a similar strategy for generating training data with the exception that corresponding map frames are selected from the same sequences. Therefore, we use sequences holding overlapping trajectories as training data for our loop closure investigations, namely 00, 02, 05, 06, and 07. In order to avoid the selection of map clouds in surrounding frames, valid candidates are chosen within a $20\,m$

radius around the current frame with a temporal distance of at least $60\,s$ resembling to 600 frames. Furthermore, we downsize source and target frames for KITTI with a voxel size of $0.2\,m$ and $0.6\,m$ respectively and use 3D poses for ground truth pose generation, so $\delta_t = \sqrt{t_x^2 + t_y^2 + t_z^2}$ but with rotation constraint to an angular offset of the yaw angle only $\delta_r = [-\pi, \pi]$.

Metrics: We measure transformation errors T between the predicted poses T_{pred} and the the respective dataset's ground truth poses T_{GT} as:

$$T = T_{\text{pred}}^{-1} \cdot T_{\text{GT}}. \tag{10}$$

Subsequently, we calculate translation errors E_t as euclidean norm in meters and rotational errors E_r as angular offset in radians (cp. [9]) formally denoted as:

$$E_t = \left\| (T_{41}, T_{42}, T_{43})^{\text{T}} \right\| \tag{11}$$

$$\begin{aligned} E_r &= \arccos f_\theta\big(0.5 \cdot (T_{11} + T_{22} + T_{33} - 1)\big) \\ f_{\theta(x)} &= \begin{cases} 1 & \text{if } x > 1 \\ x & \text{if } -1 < x < 1 \\ -1 & \text{if } x < -1 \end{cases} \end{aligned} \tag{12}$$

State-of-the-art methods: We compare our results on relocalization and loop closing to the geometry based state-of-the-art methods ICP [33] and G-ICP [22] as well as to the more complex FPFH [19] using hand-crafted feature descriptors. We utilize the point cloud library (PCL) [20] for the implementations of ICP and G-ICP and Open3D for FPFH. Furthermore, we consider state-of-the-art DNN approaches for point cloud registration, i.e., D3Feat [2] and 3DFeat-Net [30] using their released replication code. For FPFH and the DNN methods, we apply RANSAC to the calculated key-points and feature descriptors to determine the pose. Furthermore, we downsample point clouds with the voxel sizes discussed in the *Training details* paragraph for all ICP variants and for all other methods following to their original publication and code instructions.

4.2. Relocalization

Test dataset: We selected the *Boston-Seaport* test set of the nuScenes dataset to validate StickyLocalization's relocalization performance. Note that training was performed on the *Singapore-Onenorth* subset, which results in a test set of unseen frames from a completely different environment. The random pose generation for validation was executed once and then saved in order to supply the same transformations to all compared methods. In this context we aimed for equally distributed translational offsets, so $1/3$ of the frames had a displacement chosen from $\delta_{t1} = [0, 5]\,m$, $\delta_{t2} = [5, 10]\,m$, and $\delta_{t3} = [10, 20]\,m$ respectively. Based on this distribution, we derive three datasets, each having a different constraint maximum rotational offset per frame, resulting in the respective variants *Easy:* $\delta_{r1} = [-0.25\pi, 0.25\pi]$, *Medium:* $\delta_{r2} = [-0.5\pi, 0.5\pi]$, and *Hard:* $\delta_{r3} = [-\pi, \pi]$. The performance of the considered methods for pose estimation in the course of the relocalization task is shown in Tab. 1.

Discussion: The majority of the compared methods show substantial deficiencies in achieving accurate pose estimation due to the large temporal offset, of several days, between frame and map section and the accompanying cloud discrepancies. We find that the performance of the less complex approaches, i.e. ICP and G-ICP, is highly depended on the maximum angular offset of the transformations expressed by a decrease of accuracy for higher levels of difficulty of the test sets. The descriptor based methods on the other hand show an overall constant performance on the three test sets, which suggests a rotational invariance of these approaches. However, due to the presence of dynamic objects and key-points being featured on their respective surfaces, the subsequent RANSAC struggles in estimating accurate poses for the calculated/predicted feature points and descriptors. To overcome these challenging conditions, StickyLocalization utilizes *Graph Neural Networks* and makes extensive use of attention mechanisms allowing it to learn and analyze inner- and inter-cloud relations. We observe that StickyLocalization predicts higher probabilities for correspondences from the static environment of the current frame and the respective map section rather than temporal key-points originating from dynamic objects. Consequently, by applying SVD to the robust matches we are reaching the lowest mean translational and rotational errors among all considered methods for the employed relocalization test sets. Fig. 3 shows qualitative results visualizing the correspondences predicted by our approach which are subsequently used to estimate the vehicles pose inside the map.

Runtime evaluation: In the course of our experiments we, amongst others, performed tests using different parametrization for the deep learning aided methods 3DFeat-Net and D3Feat concerning the quality of the estimated poses based on the predicted key-points and features. We found that the performance is highly depending on the number of RANSAC iterations, hence we chose a comparatively high number of 4,000,000 iterations, to gain fair and comparable results on our experiments. A high amount of RANSAC iterations, however, results in an increase of computation time, constraining the applicability of an algorithm. In this context, an additional advantage of our approach is the independence of a sub-sequential fitting algorithm like RANSAC, since we are able to implicitly predict inter-cloud correspondences which can be utilized by SVD in order to predict the respective poses. Since SVD is more efficient, StickyLocalization outperforms all compared methods in terms of runtime by

Table 1. **Relocalization results** on the nuScenes test data (*Boston-Seaport*)

Method	$\delta_t[0;5]m$		$\delta_t[5;10]m$		$\delta_t[10;20]m$		mean	
	E_t	E_r	E_t	E_r	E_t	E_r	E_t	E_r
Easy: $\delta_r \leq 45$ deg								
ICP[33]	1.848	0.151	3.000	0.144	7.307	0.191	4.084	0.162
G-ICP[22]	1.701	0.159	3.395	0.166	8.897	0.235	4.707	0.187
FPFH[19]	4.065	0.136	4.068	0.141	4.743	0.143	4.297	0.140
3DFeat-Net[30]	1.660	0.068	1.501	0.058	1.611	0.088	1.591	0.072
D3Feat[2]	0.734	0.035	0.760	0.051	1.280	0.072	0.928	0.053
Ours	**0.561**	**0.017**	**0.387**	**0.012**	**0.517**	**0.019**	**0.489**	**0.016**
Medium: $\delta_r \leq 90$ deg								
ICP[33]	4.125	0.579	6.207	0.637	9.345	0.628	6.587	0.615
G-ICP[22]	4.341	0.593	6.547	0.652	10.955	0.665	7.318	0.637
FPFH[19]	5.318	0.288	6.059	0.308	6.840	0.262	6.080	0.286
3DFeat-Net[30]	1.798	0.091	1.896	0.069	1.672	0.065	1.788	0.075
D3Feat[2]	0.762	0.045	0.958	0.059	0.998	0.045	0.907	0.049
Ours	**0.431**	**0.011**	**0.317**	**0.008**	**0.605**	**0.015**	**0.452**	**0.011**
Hard: $\delta_r \leq 180$ deg								
ICP[33]	6.726	1.533	7.735	1.508	11.019	1.451	8.518	1.497
G-ICP[22]	6.131	1.494	8.321	1.505	13.569	1.463	9.383	1.487
FPFH[19]	5.986	0.384	6.386	0.366	8.166	0.427	6.859	0.392
3DFeat-Net[30]	1.810	0.085	1.836	0.102	2.008	0.107	1.886	0.098
D3Feat[2]	1.079	0.029	0.947	0.048	1.208	0.040	1.080	0.039
Ours	**0.553**	**0.010**	**0.355**	**0.009**	**0.401**	**0.014**	**0.436**	**0.011**

a substantial margin. The average runtime per approach for matching one pair of frames on the nuScenes test set is shown in Fig. 2.

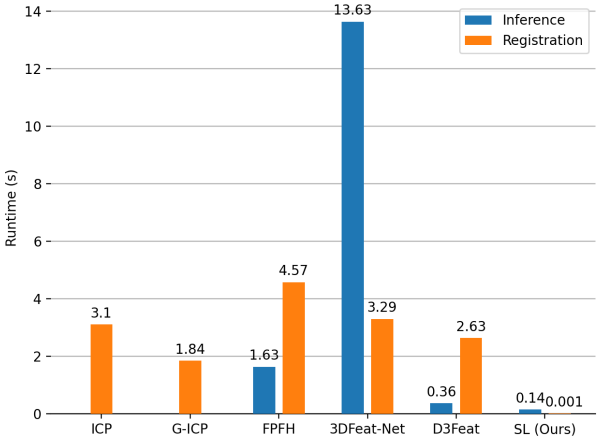

Figure 2. **Average runtime** across the compared methods on the *Hard* variant of the nuScenes test dataset. The ability to use SVD for registration in our approach presents a huge runtime benefit in contrast to methods relying on RANSAC.

4.3. Loop Closure Estimation

Test Dataset: For the loop closure estimation experiments we utilize sequence 08 of the KITTI dataset, which was unseen during the training of our network. We refined the ground truth poses for each frame pair, as these are known to feature certain drifts. Furthermore, we again construct three different test sets with varying δ_r and δ_t analog to the previous experiments. Translational displacement generation in this case, however, was done in a three-dimensional manner so that $\delta_t = \sqrt{t_x^2 + t_y^2 + t_z^2}$ with rotational offset constraint to the yaw angle, since this is the most prominent angle for a vehicle's poses. Table 2 reports the performance of our approach in contrast to the considered state-of-the-art methods on the respective test sets.

Discussion: Aside from the ICP variants, the considered methods provide results with substantially lower overall transformation errors in contrast to the relocalization experiments. This is due to frame pairs being part of the same trace and therefore not comprising discrepancies caused by dynamic objects. Larger errors concerning ICP and G-ICP on those experiments are most likely caused by the general rotation variance of the approaches in combination with the more dense point clouds delivered by the Velodyne HDL-64E used for the KITTI dataset. StickyLocalization still reaches the lowest rotational errors on all test sets as well as comparable results to the deep learning based state-of-the-art approaches with respect to the average translational errors. We argue that due to missing discrepancies between the point clouds for the loop closure estimation task, RANSAC converges much faster for D3Feat.

Table 2. **Loop closure estimation results** on the KITTI test data (*Sequence 08*)

Method	$\delta_t[0;5]m$		$\delta_t[5;10]m$		$\delta_t[10;20]m$		mean	
	E_t	E_r	E_t	E_r	E_t	E_r	E_t	E_r
Easy: $\delta_r \leq 45$ deg								
ICP[33]	4.807	0.208	4.562	0.155	4.577	0.233	4.648	0.199
G-ICP[22]	2.139	0.175	2.584	0.093	4.311	0.463	3.026	0.246
FPFH[19]	1.379	0.100	0.749	0.047	0.680	0.069	0.956	0.072
3DFeat-Net[30]	0.785	0.034	0.569	0.034	0.659	0.018	0.674	0.029
D3Feat[2]	**0.352**	0.017	**0.348**	0.017	0.403	0.017	**0.366**	0.017
Ours	0.392	**0.009**	0.381	**0.009**	**0.372**	**0.010**	0.382	**0.010**
Medium: $\delta_r \leq 90$ deg								
ICP[33]	7.361	0.610	7.608	0.704	7.541	0.625	7.504	0.646
G-ICP[22]	5.357	0.641	5.784	0.731	7.855	0.842	6.350	0.739
FPFH[19]	1.299	0.108	2.730	0.206	1.425	0.065	1.823	0.128
3DFeat-Net[30]	0.431	0.017	0.347	0.013	0.609	0.037	0.458	0.022
D3Feat[2]	**0.319**	0.018	**0.342**	0.017	0.426	0.017	**0.360**	0.017
Ours	0.400	**0.010**	0.375	**0.010**	**0.378**	**0.010**	0.385	**0.010**
Hard: $\delta_r \leq 180$ deg								
ICP[33]	9.424	1.441	10.174	1.669	10.864	1.588	10.162	1.566
G-ICP[22]	7.502	1.421	8.043	1.658	10.124	1.756	8.575	1.614
FPFH[19]	2.518	0.217	3.149	0.242	3.092	0.207	2.903	0.222
3DFeat-Net[30]	0.797	0.057	0.754	0.038	0.809	0.058	0.787	0.051
D3Feat[2]	**0.344**	0.016	0.387	0.018	0.384	0.017	**0.371**	0.017
Ours	0.456	**0.010**	**0.354**	**0.009**	**0.382**	**0.010**	0.399	**0.010**

4.4. Ablation Study

In order to study the impact of the individual contributions proposed in this paper on the overall performance of our approach, we perform continuative trainings while altering certain parts of the processing pipeline. More precisely, we compare four different versions of StickyLocalization and evaluate their respective performance on the Hard set of nuScenes test data, i.e.:

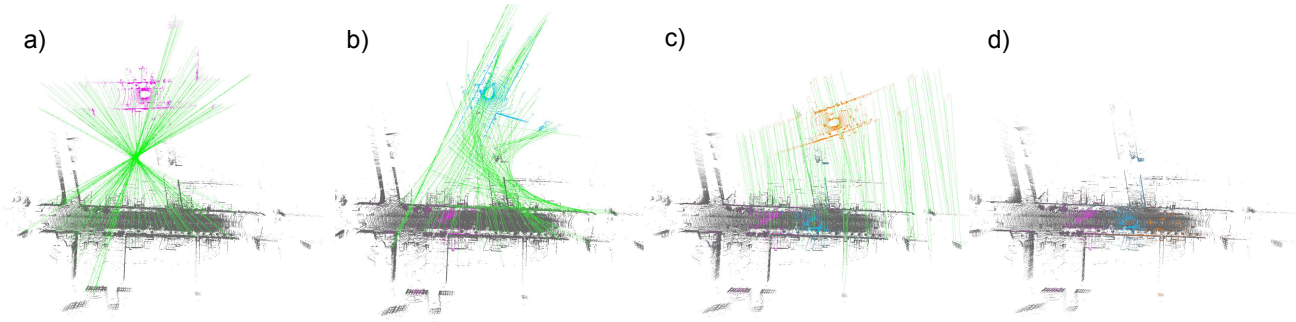

Figure 3. **Qualitative results:** Current measurements (colored) and the corresponding map (gray) used for relocalization on the nuScenes test dataset. Green lines highlight the correspondences predicted by StickyLocalization used to determine the vehicles poses inside the map. The estimated poses are subsequently applied to the respective point clouds to achieve an overlay with the map.

- SL_{PS} : Using preselected features based on the smoothness factor proposed by [32] and the loss function based on hard matches (Eq. 8)
- SL_{RND} : Using randomly sampled points, trained with the hard matches loss (Eq. 8)
- SL_{HM} : Trained with learnable feature selection but applying the hard matches loss (Eq. 8)
- SL_{DM} : Trained with learnable feature selection and applying the proposed distance matching loss (Eq. 9)
- SL_{RSC} : Model SL_{DM} but applying RANSAC directly on the predicted descriptors

Table 3. **Ablation experiments** with different configurations of StickyLocalization on the nuScenes test data

Method	$\delta_t[0;5]m$		$\delta_t[5;10]m$		$\delta_t[10;20]m$		mean	
	E_t	E_r	E_t	E_r	E_t	E_r	E_t	E_r
SL_{RND}	1.382	0.167	1.071	0.101	0.887	0.085	1.111	0.117
SL_{PS}	0.795	0.068	0.921	0.063	0.956	0.063	0.892	0.065
SL_{HM}	1.254	0.085	0.932	0.042	1.376	0.109	1.189	0.079
SL_{DM}	0.553	**0.010**	**0.355**	**0.009**	0.401	**0.014**	0.436	**0.011**
SL_{RSC}	**0.545**	0.020	0.361	0.014	**0.394**	**0.014**	**0.433**	0.016

Table 3 reports the performance of the different configurations. Applying the inline feature selection method (cp. Sec. 3.1) and the distance based matching loss (cp. Eq. 9), results in lower transformation errors compared to the configurations using pre-selected features or random key-points, demonstrating StickyLocalization's ability to learn a robust selection of significant key-points for pose estimation. Furthermore, in comparison to using a loss based on hard matches in the course of joint training of key-point selection and matching, our proposed loss leads to faster convergence of the network yielding lower transformation errors.

Finally we validate the quality of the descriptors predicted by the *Graph Neural Network* by directly feeding them into RANSAC to predict the pose. Here, we utilized the model SL_{DM} but skipping the Sinkhorn normalization and SVD for inference time on the test data. Because of the included outlier removal of RANSAC, partially small improvements regarding E_t can be observed compared to SL_{DM}. Due to skipping the Sinkhorn iterations, we see a drop in inference time from $0.140s$ to $0.080s$, but simultaneously the average registration time increases from $0.001s$ to $0.181s$. However, the performance of RANSAC in these experiments proves the high expressiveness of the descriptors predicted by our *Graph Neural Network* and their potential utilizability for other perception tasks.

5. Conclusion

In this paper, we present a fast and accurate method for point cloud based relocalization utilizing *Graph Neural Networks*. By introducing a descriptor-based feature pre-selection and utilizing a novel distance based matching loss, we are able to construct an end-to-end trainable pipeline for joint robust key-point detection and matching. To demonstrate our performance, we performed experiments using sequences of the nuScenes dataset recorded in a completely different area compared to the sequences used to train the network. We compare our results to traditional and deep learning based state-of-the-art methods and are able to outperform all approaches in terms of transformation errors and runtime. Furthermore, we evaluated the applicability of StickyLocalization with regard to loop closure estimation tasks, where we employed the training sequences of KITTI including overlapping odometry traces. Again, StickyLocalization outperforms all compared methods concerning rotational errors and achieves state-of-the-art results in terms of translational. Finally we showed that by utilizing the stated selection strategy in combination with the proposed distance based matching loss, we achieve the best performance among all versions of our approach with certain parts replaced, as well as the overall quality of our descriptors by directly applying RANSAC in order to predict the pose.

References

[1] Mikaela Angelina, Uy Gim, and Hee Lee. PointNetVLAD : Deep Point Cloud Based Retrieval for Large-Scale Place Recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–2, 2018.

[2] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6359–6367, 2020.

[3] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, Feb. 1992.

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11621–11631, 2020.

[5] Changhao Chen, Bing Wang, Chris Xiaoxuan Lu, Niki Trigoni, and Andrew Markham. A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence. *arXiv preprint arXiv:2006.12567*, 2020.

[6] Philipp Egger, Paulo V.K. Borges, Gavin Catt, Andreas Pfrunder, Roland Siegwart, and Renaud Dube. PoseMap: Lifelong, Multi-Environment 3D LiDAR Localization. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3430–3437, 2018.

[7] Nico Engel, Stefan Hoermann, Markus Horn, Vasileios Belagiannis, and Klaus Dietmayer. DeepLocalization: Landmark-based Self-Localization with Deep Neural Networks. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 926–933, 2019.

[8] Kai Fischer, Martin Simon, Florian Oelsner, Stefan Milz, Horst-Michael Gross, and Patrick Maeder. Stickypillars: Robust and efficient feature matching on point clouds using graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 313–323, 2021.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[10] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-dof camera relocalization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946, 2015.

[11] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[12] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-based precision vehicle localization in urban environments. *Robotics: Science and Systems*, 3:121–128, 2008.

[13] Jesse Levinson and Sebastian Thrun. Robust Vehicle Localization in Urban Environments Using Probabilistic Maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4372–4378, 2010.

[14] Jiarong Lin and Fu Zhang. Loam_livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov. *arXiv preprint arXiv:1909.06700*, 2019.

[15] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. Deepvcp: An end-to-end deep neural network for point cloud registration. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[16] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS Workshops*, 2017.

[17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[18] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001.

[19] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[20] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4. IEEE, 2011.

[21] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. *arXiv preprint arXiv:1911.11763*, 2019.

[22] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[23] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[25] Victor Vaquero, Kai Fischer, Francesc Moreno-Noguer, Alberto Sanfeliu, and Stefan Milz. Improving Map Relocalization with Deep 'Movable' Objects Segmentation on 3D LiDAR Point Clouds. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 942–949, 2019.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[27] Wei Wang, Bing Wang, Peijun Zhao, Changhao Chen, Ronald Clark, Bo Yang, Andrew Markham, and Niki Trigoni. Pointloc: Deep pose regressor for lidar point cloud localization. *arXiv preprint arXiv:2003.02392*, 2020.

[28] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3523–3532, 2019.

[29] Ryan W. Wolcott and Ryan M. Eustice. Fast LIDAR localization using multiresolution Gaussian mixture maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2814–2821, 2015.

[30] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *European Conference on Computer Vision*, pages 630–646. Springer, 2018.

[31] Huan Yin, Li Tang, Xiaqing DIng, Yue Wang, and Rong Xiong. LocNet: Global Localization in 3D Point Clouds for Mobile Vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 728–733, 2018.

[32] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Proceedings of Robotics: Science and Systems Conference*, July 2014.

[33] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.